EED-2 Open Sourcing Guide

A mostly comprehensive resource for those wanting to publish projects in the ECC onto GitHub

Github has an article that provides a pretty good overview on removing sensitive information, which can be found here:

https://help.github.com/articles/removing-sensitive-data-from-a-repository/

On Searching

There are likely many methods by which you can search through a repository for sensitive information. Below is a fairly straightforward example of one.

Clone the repository and track all branches:

```
git clone MY_GIT_REPOSITORY.git

cd MY_GIT_REPOSITORY

for branch in `git branch -a | grep remotes | grep -v HEAD | grep -v master`; do

git branch --track ${branch##*/} $branch

done
```

- Search repository for sensitive patterns (tokens, passwords, hostnames, IP's, etc)
- git rev-list --all | git grep -E '\b(YOUR REGEX HERE)\b' >> ~/found_items.txt

This is arguably the trickiest part of the whole thing.

Sites like http://regex101.com have great regex testers if you're as rusty with them as I am, though the hardest part is knowing what to search for, and unfortunalely I can't help you with that.

On Scrubbing

There are two methods for repository scrubbing. One leverages git filter-branch, and the other uses the BFG Repository cleaner. Both will work, but BFG works faster, and both the CMR and MMT teams have had success with it.

- BFG Repo Cleaner https://rtyley.github.io/bfg-repo-cleaner/
  - Great for removing sensitive files and redacting things like tokens and passwords from config files.
  - It's worth noting that BFG assumes your latest commit is a good one.
  - **If you want a file gone you must have a commit which deletes the file. BFG assumes that anything in your HEAD is the way it should be.**
  - BFG has two flags that allow you to remove files/directories. Removing directories is exactly as you would expect, but the --rem ove-files only takes a single file, and it only takes the file _name_ (no paths allowed, not even relative ones)
  - assuming you've deleted files you want gone and are ready to have them erased run the following before staging/commiting the m: git status --porcelain | sed -e 's!.*/!!' >> path/to/files_to_delete.txt
  - If you have a lot of files to delete, running BFG for each file is going to be a pain, so use the bash snippit below instead!
  - If you don't want an empty commit as your new HEAD, try this: git reset --soft "HEAD^" then this: git commit --amend. After a forc e-push, nobody will be the wiser!
  - GitHub is often picky about file and repository size. When running the BFG repo cleaner, you may need to use this option: `--str ip-blobs-bigger-than 100M`

- git filter-branch
  - A resource for removing passwords from files: http://www.davidverhasselt.com/git-how-to-remove-your-password-from-a-reposit ory/
  - The rest of the process is outlined in sufficient detail in the Github article posted above, but to save the trouble of scrolling, it's been added again below.
  - https://help.github.com/articles/removing-sensitive-data-from-a-repository/

Regardless of which option you choose, you'll need to garbage collect! It is useful to have another empty private repository ready to push your scrubbed and garbage-collected code to for verification. Garbage collection is done with the following two commands:

```
git reflog expire --expire=now --all && git gc --prune=now --aggressive
```

Bash snippit for automating BFG on many files. If running as a .sh file, don't forget to run chmod +x <filename> to give execution permission

```
#! /bin/bash
IFS=$'\n' read -d '' -r -a files < {PATH TO FILE}/files_to_delete.txt
for i in "${files[@]}"; do
 java -jar {PATH TO BFG JAR} --delete-files $i
done
```

Rebasing currently in-progress work onto the squeaky-clean new repository

Once you've finished scrubbing and pushed to the repository, it's important that anybody with ongoing work follow the relevant steps in the article below to ensure that they don't dirty the history.

Merge commits from a dirty base will undo all of the scrubbing that you just did!!!!

- http://blog.ostermiller.org/git-remove-from-history
    - Step 8 details how to ensure that nobody accidentally dirties the history. This is essential to make sure nobody undoes your scru bbing!